

Parcours de labyrinthe

Le problème est le suivant. Vous êtes dans un labyrinthe. L'entrée se trouve, par exemple, en haut à gauche, en (1,1), et la sortie en bas à droite. On veut à partir de l'entrée trouver un chemin qui mène à la sortie. Quatre directions sont autorisées : haut, bas, droite et gauche. Une solution est figurée avec des points.

```

. #####
.# ##...  #
.. # .##. # ###
#..... #.....
# # #####.#
#####.....#
#      .#####
##### .....

```

Un algorithme pour le labyrinthe

Construire un chemin libérateur dans le labyrinthe repose sur un algorithme de *backtracking*. On se place à l'entrée du labyrinthe, en (1, 1), et on essaye la première direction autorisée. Disons, à droite. On se place en (2, 1) et on stocke l'information (2, 1) dans une pile. À partir de (2, 1), on essaye toutes les directions possibles, par où le chemin n'est pas déjà passé : il n'y en a aucune. Le choix (2, 1) était donc un mauvais choix, que l'on remet en cause en dépilant (2, 1). De (1, 1), on essaye maintenant la direction autorisée suivante, vers le bas, en (1, 2). On empile (1, 2). De (1, 2), la première direction autorisée est de nouveau vers le bas, en (1, 3). On empile (1, 3). De (1, 3), la première direction autorisée est vers la droite. On empile (2, 3), et ainsi de suite. Si on arrive à une position bouchée, on remonte dans le chemin en dépilant un élément, et on essaye la direction suivante. Le procédé s'arrête quand on a trouvé la sortie, ou quand la pile est vide. Dans ce dernier cas, on a essayé en vain tous les chemins, et le labyrinthe est sans issue.

Implémentation

Un squelette de programmation pour ce problème est disponible dans le répertoire

```
~/touzet/Algo/Labyrinthe
```

Les fichiers `paq_pile.ad*` contiennent un paquetage pour la gestion de piles dont les éléments sont des positions dans le labyrinthe. Le fichier `laby.adb` concerne la construction d'un chemin dans un labyrinthe. Il contient d'ores et déjà la définition des types `Labyrinthe` et `Position`, ainsi que les fonctions `Afficher` et `Initialiser`. Les fichiers `lab1`, `lab2`, ... sont des fichiers textes, avec des exemples que vous pourrez utiliser pour tester le programme.

La fonction `Afficher` permet d'afficher un labyrinthe avec un chemin stocké sous forme de pile.

La fonction `Initialiser` lit un labyrinthe dans un fichier texte, et le charge dans un tableau de type `Labyrinthe`. Une manière simple de délimiter les contours du labyrinthe est d'ajouter une rangée de murs en bas, en haut, à gauche et à droite. C'est-à-dire qu'un labyrinthe de 4×4 est représenté par un tableau de 6×6 . C'est ce que fait la fonction `Initialiser`.

Votre travail

Compléter la procédure `Trouver_chemin` qui construit un chemin libérateur de `Entree` à `Sortie` dans le labyrinthe `L` et le mémorise dans `Chemin`. Pour cette procédure, pensez à définir, en plus du labyrinthe, un second tableau de booléens pour mémoriser les cases par lesquelles vous êtes déjà passé.